Assessing Opinion Mining in Stock Trading

Sathish Nagappan (srn), Govinda Dasu (gdasu)

I. Introduction

We hypothesize that money should go where the public wants and needs it to go, and the firm that is the best and fastest at determining these human demands will yield the highest percentage growth. To test this hypothesis we set up two predictors: (1) the first predictor considered only numerical data such as historical prices, EBITDA, and PE Ratio, and (2) the second considered human news and opinions on companies, their products, and their services. The first task was to derive an accurate first predictor. As our baseline we used SVR with RBF kernel, which led to SGD with various iterations to better approximate the RBF kernel. After implementing this by grouping all stocks from major indices, we realized that we should consider stocks individually and take into account time series. This resulted in the ARIMAX model with AIC backwards search selection (predictor 1). Next, we moved to predictor 2. We added NLP features for each company such as indicators of specific n-grams that give insight into the positivity of the stream of relevant text about a company's products and services. Ultimately this led to ARIMAX with these NLP features and combination feature selection (predictor 2). This allowed us to compare the relative successes of the model with and without NLP features.

II. Data and Cross Validation

The numerical data was obtained from Bloomberg and the headline data from Factset. We retrieved a list of all stocks from the S&P 500, Russell 1000, and NASDAQ 100. Each training example was indexed by company ticker and date and had 28 features such as PE Ratio, EBITDA, price, and volatility. The target for each training example was the one day percent change in closing price. We retrieved 2M training examples from 2009 to present. Our method of evaluation comes from the concept of score defined as follows:

$$Score = 1 - \sum_{i=1}^{m} (y(i)_{true} - y(i)_{predicted})^2 / \sum_{i=1}^{m} (y(i)_{true} - y(i)_{true\_mean})^2$$

Perfect prediction yields a score of 1. Less optimal predictions will be lower, even arbitrarily large negative numbers. We used a variant of hold-out cross validation; we tested our models on the last 6 months and trained using the remainder of the data. Due to computational complexity, for our initial algorithm, we trained on the first 1.5 years of 2012-2013 and tested on the last 6 months, totaling 486k training examples. For our later algorithms, we trained on the first 3.5 years of 2009-2013 and tested on the last 6 months. Randomly selecting a subsample to cross validate would yield an unrealistic and unfair advantage since we would be using future prices to predict tomorrow's prices. In addition, market data but not computational power was abundant, justifying the use of hold-out cross validation as opposed to k-fold validation (Blum et. al, 1999).

III. Baseline System

Literature suggested that SVMs with RBF (radial basis function) kernel are natural choices for the stock trading problem (Bao et. al, 2004) and (Lee, 2009).

SVR with RBF Kernel: Since we are dealing with regression (output of percent change in stock price), we explored SVR (support vector regression). We quickly found the algorithm was too computationally expensive for our 68 MB memory EC2 instance. The implementation of SVR in scikit-learn is $O(n*m^3)$, which was not scalable given our large dataset. We were only able to efficiently learn on 10k training examples, a small fraction of our 486k examples. To combat this issue, we subsampled 10k training examples over the years 2012-2013. Although our algorithm was now executing, we ran into a variance problem; our test score was -0.01812 and our training score was 0.02646. It was clear we needed a computationally simpler approach so we could make the most out of all of our data.

IV. Stochastic Gradient Descent (SGD)

We proceeded through a number of iterations to improve our baseline system.

SGD with RBF Approximation: We implemented SGD with shuffling and used an RBF approximation to kernelize the input. Our literature review suggested that this was a good approximation for SVR with RBF kernel (Hazan et. al, 2011). We approximated the RBF kernel using Nystrom's method. Nystrom's method is parameterized by the number of Monte Carlo samples, n, which corresponds to the dimensions of the feature vector (Rahimi and Recht, 2007). We implemented a model selection scheme to loop through n = [1, 65] and γ = [.1,1] where the upper bound on n and lower bound on γ (parameter for RBF) were set due to memory constraints, and we

achieved unsatisfying results of test score -3.0339e-4 and training score 1.4043e-4 with optimal values of γ=.4 and n=35. Interestingly, the parameter values had a negligible impact on our results. It became clear to us that Nystrom's method was not approximating well for our dataset (Affandi et. al, 2013).

SGD with RBF Approximation with K-Means: To improve our approximation of the RBF kernel, we drew inspiration from Zhang et. al (2008). In their paper, they describe a method of using the clusters from K-means as the basis for Nystrom. We implemented this algorithm. We achieved better results of test score 1.9061e-4 and training score 3.2344e-5 with optimal values of γ=.8 and n=16. K-means was computationally expensive so we could only compute a maximum of n=20, which corresponds to the number of clusters in K-means and Monte Carlo samples. Encouragingly, however, as we increased n, unlike in the prior case, our prediction accuracy improved as expected (Drineas et. al, 2006). This led us to believe that we needed a more efficient way to compute K-means so that we could operate on higher n values.

SGD with RBF Approximation with K-Means Mini-Batch: Tasked with trying to compute K-means more efficiently, we found the results from Scully (2010). The author proposed a K-means mini-batch algorithm with complexity orders of magnitude lower than that of the original K-means algorithm. This significantly boosted the value for n that we could use. We looped through n = [1, 65] and γ = [.1,1] where the bounds were set to parallel the original SGD algorithm, and we achieved significantly better results of test score 5.9165e-4 and training score 5.1006e-4 with optimal values of γ = .18 and n=61.

Failure Points: Although our results for the K-means mini-batch algorithm were significantly better than our original attempt, our results were still unsatisfying. Our model suffered from a bias problem, and the proximity of the test and training scores verified that. To solve this bias problem, we needed to improve our feature set by addressing three flaws: lack of time-series, technical trading, and collective predictions. Our model had no notion of time. Every training example, regardless of the day, was treated equally. Current market trends and previous day prices have a significant impact on predicting prices for tomorrow (LeBaron et. al, 1999). As a result, we investigated the autoregressive integrated moving-average (ARIMA) model. Secondly, technical trading does not yield good prediction results because we are only taking into account the quantitative and not qualitative factors that influence stock prices (Brown and Cliff, 2004). To this end, we built a sentiment analyzer. Finally, we grouped all the stocks together to make predictions. We feel we would get better results if we look at each stock individually as explained by Atsalakis and Valavanis (2009).

V. Predictor 1: ARIMAX with AIC Backwards Search Selection

We replaced SGD with the ARIMA (auto-regressive integrated moving average) model to take time-series into account. By using ARIMA, we take into account the fact that financial stocks are best modeled as weakly stationary stochastic processes (Pai and Lin, 2005). In other words, financial stocks move from one price to another with some probability which is given by some underlying probability distribution. The following is the equation for the ARMA model:

$$y(t) = \sum_{i=1}^{m} a(i) \cdot y(t - i) + \sum_{i=1}^{n} b(i) \cdot x(t - i)$$

The first summation represents the autoregressive (AR) component and the second summation represents the moving average (MA) component. The MA component is a function of the inputs *x(t - i)*. Moreover, *x(t - i)*, the input, is in turn a function of the white noise of our stochastic system. In this way, the white noise is passed through both a recursive (AR) and non-recursive filter (MA).

We used the ARIMAX model which is a generalization of the ARMA model. The ARIMA models includes an integrated component, and the ARIMAX model allows multiple exogenous variables. The order is given by (p, d, q) where p is the AR term, d is the integrated term, and q is the MA term.

We built our model around a single stock. The coming sections will provide an in-depth analysis on the Google stock (GOOG). Note however, we did not observe significant differences between the stocks we chose. At the end, we will present final results for two other stocks: Abbott Laboratories (ABT) and General Electric (GE). We chose these three stocks since they are in three different industries. We trained on the first 3.5 years of 2009-2013 and tested on the last 6 months.

Order Selection: To determine the order of our ARIMA model, we used grid search on $0 \le p, d, q \le 6$ and we achieved an optimal order of $(p, d, q) = (1, 0, 6)$. Encouragingly this was consistent to the results found by Nelson for asset returns (1991). To verify that this order accurately represented the data and was not an artifact of grid search, we used the analysis and techniques from Ozaki (1977). Ozaki describes a method of using autocorrelations and partial autocorrelations for selecting orders. We did preliminary calculations to verify

that our calculated order was reasonable. Unfortunately, despite this, our computed score was worse. We got a training score of 0.03187 and test score of -0.04153. The discrepancy between the training score and test score led us to believe that we were suffering from a variance issue.

Feature Selection on Original Features: To combat this variance problem, we decided to use feature selection to remove unnecessary features. Since we have a reasonably small feature set, we used the backwards search algorithm. To evaluate our models, we used a similar approach to that of Tsay (1980) and used Akaike information criterion (AIC). AIC is a commonly used metric for ARIMA model selection and represents a tradeoff between the fit and model complexity. This approach eliminated six features: bid price, 200 day moving average, 50 day moving average, high price, diluted EPS, and total shares outstanding. Interestingly feature selection decided to remove moving average which was a core problem of the SGD with k-means mini-batch algorithm. This could be due to the fact that ARIMA inherently takes into account moving average so these two features are unnecessary. Our score, training score, and AIC values were 0.0138, 0.0203, and 4471.8 respectively. We achieved almost two orders of magnitude improvement from SGD with RBF kernel approximated with k-means mini-batch.

VI. Predictor 2: System with Opinion Mining and Combination Feature Selection

We improved the ARIMA model's feature set by implementing opinion mining on trending news headlines and filings for public companies. We selected 3 stocks (GOOG, ABT, and GE) to run our improved system on and treated the stocks separately, not allowing, say, JP Morgan's data to influence our predictions on Google's stock. The 93 additional features that we introduced were:

- indicators of whether a particular headline had instances of the entire corpus's 30 most common unigrams, bigrams and trigrams (weighted by the inverse depth of the n-gram in the parse tree of the headline)
- the length of a headline
- the maximum parse depth of a headline
- the proportion of the words of the headline that were adjectives and adverbs

Selection on NLP Features: The ARIMA model on the original plus NLP features yielded a score of -0.02172 and training score of 0.09110. The large difference between the test score and training score made it clear that we were dealing with a variance problem. We quickly ran into a problem with our AIC backwards search algorithm; this algorithm is $O(n^2)$ time complexity and the large increase in features made it unusable for our problem. We first tried principal component analysis (PCA). This did not work for us as we got negative training and test scores. We did research and found the results of Huang and Tsai (2009). They implemented a filter-based feature selection algorithm that had promising results for stock market predictions. We refer the reader to Huang and Tsai's paper for details on the algorithm. Their algorithm too, however, yielded even worse results than no feature selection algorithm. It was clear we needed a better way to select features which led us to our next algorithm.

PCA and Filter-Based Feature Selection on NLP Features: Tsai and Hsiao researched whether using multiple feature selection algorithms to jointly provide predictions would provide better results than using a single feature selection algorithm in the context of stock predictions (2010). Their best performing method used PCA and genetic algorithms (GA). They used union, intersection, and multi-intersection approaches to select features. It is fruitless to reproduce all of the details and theory of their procedure in this paper, and we refer the reader to Tsai and Hsiao's paper for details. We implemented a variant of this method. We used PCA and Huang and Tsai's filter feature selection method. This algorithm improved our results dramatically; we removed 19 features and our test score was .05975 and training score was .06262.

VII. Opinion Mining Discussion Error Analysis

The biggest question of our project was whether opinion mining would have a significant positive effect on stock price change predictions and our results answer a resounding yes. The addition of 93 NLP features (top 30 unigrams, top 30 bigrams, top 30 trigrams, Proportion of Adjectives and Adverbs, Headline Length, and Headline Parse Depth) improved our score from 0.01383 to 0.05975 which is a 332% improvement due to opinion mining and combination feature selection as described above.

Effect of Featurizing n-gram Depth: Note that we embedded information about parse tree depth in our featurization of n-grams for each headline. In order to test whether depth information actually had an effect and moreover a positive effect on our generalization score, we ran one trial on the GOOG stock where we removed depth information and simply stored indicator features of whether or not certain unigrams, bigrams, and trigrams occurred in a training example. We called this our *depthless system*. The depthless system reported a generalization score of 0.02789 and training score of 0.06601, which means that adding depth resulted in a 114% improvement. So we concluded that featurizing depth (particularly 1/depth) on top of unigrams, bigrams, and trigrams was just as important as indicating

whether those unigrams, bigrams, and trigrams occurred in the first place. This goes to show that the location of particular n-grams in the structure of a headline has a significant impact on the overall label for the headline.

Effect of Headline Length Feature: The headline length feature had a small positive effect on the score providing an improvement of 2.354e-4 or 0.3811%.

Effect of Adjective/Adverb Proportions Feature: Consistent with literature, this feature had a significant positive effect on the generalization score providing an improvement of 3.475e-3 or 5.9361%.

Effect of Headline Parse Tree Depth Feature: The headline parse tree depth feature had a small positive effect on the score providing an improvement of 2.992e-4 or 0.4848%.

VIII. Final Results

METHOD	TEST SCORE	TRAINING SCORE
Baseline: SVR with RBF Kernel	-1.8117e-2	2.6461e-2
SGD with RBF approximation	-3.0339e-4	1.4043e-4
SGD with RBF approximated with k-means	1.9061e-4	3.2344e-5
SGD with RBF approximated with K-means Mini-Batch	5.9165e-4	5.1006e-4
ARIMA on Original Features	-4.1527e-2	3.1866e-2
ARIMA with AIC Backwards Search	1.3828e-2	2.0297e-2
ARIMA with Original + NLP Features	-2.1723e-2	9.1099e-2
ARIMA + NLP + Combination Feature Selection	**5.9754e-2**	**6.2621e-2**

As promised, here are final results for ABT and GE.

METHOD	TEST SCORE	TRAINING SCORE
ABT: ARIMA with AIC Backwards Search	1.7975e-2	1.9121e-2
ABT: ARIMA + NLP + Combination Feature Selection	**7.0928e-2**	**7.3630e-2**
GE: ARIMA with AIC Backwards Search	5.7561e-3	6.1826e-3
GE: ARIMA + NLP + Combination Feature Selection	**4.6410e-2**	**5.1316e-2**

The final and best models are bolded. Notice that there were similar factors of improvement between all three stocks. We achieved a significant improvement from our baseline of 7.7871e-2 for GOOG.

IX. Future Work

Although we achieved very promising results with our final model, we feel there are additional improvements we could make. In all three stocks we achieved similar training and test scores which implies bias. Thus, plausible next steps would address improvements to ARIMA or a larger or different set of NLP features. The suggestions below address these as well as additional considerations.

Trading Strategy: Our problem concerned whether or not we could accurately predict percent change in prices of stocks. It does not deal with an actual trading strategy. For example, if our model were to predict a 5% increase in price, one would want to invest more than if our model were to predict a .1% increase. A possible next step would be to devise a trading strategy to maximize returns with this

model.

ARIMA-GARCH: The ARIMAX model represents the conditional mean of a process whereas the generalized autoregressive conditional heteroskedastacity (GARCH) model represents the conditional variance. These two models can be used in conjunction to form the ARIMA-GARCH model where ARIMA models the mean and GARCH models the variance. Awartani and Corradi did an interesting analysis for predicting the volatility of the S&P500 using various GARCH models (2005), which we feel we can adapt to our problem for better results.

Classes of Stocks: Our model is based on a single stock. In a trading setting, it may be arduous to have to compute a model for every single stock. One possibility is to research stocks and categorize them based on some financial ratios. Then, similar to what Kendall did, we can generalize our model and build it on each of these categories (2003) as opposed to a single stock. This would make our model more practically usable.

NLP Features: In terms of NLP, we would like to implement Richard Socher's Recursive Neural Tensor algorithm to understand the structure of headlines rather than simply record the depth of n-grams in headlines. Also, we would like to look into removing irrelevant words and increasing the number of features to ~100K rather than ~100.

X. Conclusion

For predictor 1, we devised the ARIMAX with AIC backwards search selection and for predictor 2, we developed ARIMAX with combination selection that considers sentence structure, n-gram depth, and proportion of modifiers (adjectives and adverbs) as features. We found that opinion mining does indeed have a significant effect on the stock predicting problem with 332% improvement. Moreover, predictor 2 outperforms our baseline SVR by a score of 7.7871e-2 for GOOG. These results are very promising, and future research should be geared towards improving the model through ARIMA-GARCH and neural tensors and the usability through trading strategies and stock classification.

XI. References

[1] Ali Rahimi, Ben Recht. 2007. Random Features for Large-Scale Kernel Machines. **[2]** Kai Zhang, Ivor W. Tsang, James T. Kwok. 2008. Improved Nystrom Low-Rank Approximation and Error Analysis. **[3]** D. Scully. 2010. Web-Scale K-Means Clustering. **[4]** Elad Hazan, Tomer Koren, Nathan Srebro. 2011. Beating SGD: Learning SVMs in Sublinear Time. **[5]** Petros Drineas, Ravi Kannan, Michael W. Mahoney. 2006. Fast Monte Carlo Algorithms for Matrices I: Approximating Matrix Multiplication. **[6]** Raja Haz Aandi, Alex Kulesza, Emily B. Fox, Ben Taskar. 2013. Nystrom Approximation for Large-Scale Determinantal Processes. **[7]** Yukun Bao, Yansheng Lu, Jinlong Zhang. 2004. Forecasting stock prices by SVMs regression. **[8]** Ming-Chi Lee. 2009. Using support vector machine with a hybrid feature selection method to the stock trend prediction **[9]** Blake LeBaron, W.Brian Arthur, Richard Palmer. 1999. Time series properties of an artificial stock market. **[10]** Gregory W. Brown, Michael Cliff. 2002. Investor sentiment and the near-term stock market. **[11]** Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. **[12]** Avrim Blum, Adam Kalai, John Langford. 1999. Beating the hold-out: bounds for K-fold and progressive cross-validation **[13]** Selene Yue Xu. Stock Price Forecasting Using Information from Yahoo Finance and Google Trend. **[14]** Ping-Feng Pai, Chih-Sheng Lin. 2005. A hybrid ARIMA and support vector machines model in stock price forecasting. **[15]** Shiladitya Sinha, Chris Dyer, Kevin Gimpel, Noah A. Smith. 2013. Predicting the NFL Using Twitter. **[16]** Bo Pang, Lillian Lee, Shivakumar Vaithyanathan. 2013. Thumbs Up? Sentiment Classification Using Machine Learning Techniques. **[17]** Soo-Min Kim, Eduard Hovy. 2007. Crystal: Analyzing Predictive Opinions on the Web **[18]** Bo Pang, Lillian Lee. 2008. Opinion Mining and Sentiment Analysis **[19]** Lun-Wei Ku, Hsiu-Wei Ho, Hsin-Hsi Chen. 2009. Novel Relationship Discovery Using Opinions Mined from the Web. **[20]** Ann Devitt, Khurshid Ahmad. 2007. Sentiment Polarity Identification in Financial News: A Cohesion-based Approach. **[21]** Tony Mullen and Nigel Collier. Sentiment analysis using support vector machines with diverse in-formation sources. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 412–418, July 2004. Poster paper. **[22]** Casey Whitelaw, Navendu Garg, and Shlomo Argamon. Using appraisal groups for sentiment anal- ysis. In Proceedings of the ACM SIGIR Conference on Information and Knowledge Management (CIKM), pages 625–631. ACM, 2005. **[23]** Daniel B. Nelson. 1991. Conditional Heteroskedasticity in Asset Returns: A New Approach. **[24]** T. Ozaki. 1977. On the Order Determination of ARIMA Models. **[25]** Ruey S. Tsay. 1984. Order Selection in Nonstationary Autoregressive Models **[26]** Cheng-Lung Huang, Cheng-Yi Tsai. 2009. A hybrid SOFM-SVR with a filter-based feature selection for stock market forecasting. **[27]** Chih-Fong Tsai, Yu-Chieh Hsiao. 2010. Combining multiple feature selection methods for stock prediction: Union, intersection, and multi-intersection approaches. **[28]** G. Atsalakis, K. Valavanis. 2009. Surveying stock market forecasting techniques - Part II: Soft computing methods. **[29]** B. Awartani, V. Corradi. 2005. Predicting the volatility of the S&P-500 index via GARCH models: the role of asymmetries. **[30]** G. Kendall. 2003. A multi-agent based simulated stock market - testing on different types of stocks.